CLAIMS

What is claimed is:

1. A machine-readable medium having stored thereon sequences of instructions which, when executed by a processor, cause the processor to perform the acts of:

disabling access to at least a first section of code in a network driver interface, wherein the network driver interface provides for communication between one or more media access control units and one or more protocol drivers in a computer system according to a set of bindings;

patching the first section of code to cause the insertion of a rerouting driver into the one or more communication paths provided by the set of bindings; and

re-enabling access to the patched first section of code.

2. The machine-readable medium of claim 1 wherein the patching is static patching.

3. The machine-readable medium of claim 2 wherein the static patching includes inserting a template jump from the network driver interface to a template in the rerouting driver.

4. The machine-readable medium of claim 3 wherein the template jumps are inserted in the network driver interface so that a CALL instruction to the protocol driver is replaced with a JUMP to the template in the rerouting driver, the template containing the CALL instruction.

5. The machine-readable medium of claim 2 wherein the patching the first section of code creates at least one new binding between the network driver interface and the rerouting driver.

6. The machine-readable medium of claim 5 wherein the at least one new binding provides for communication between one or more media access control units and a capturing unit in the rerouting driver.

7. The machine-readable medium of claim 6 wherein the capturing unit is used to intercept communications over the at least one new binding.

8. The machine-readable medium of claim 1 wherein the patching is dynamic patching.

9. The machine-readable medium of claim 8 wherein the dynamic patching includes establishing a new binding between at least one media access control unit and dynamic patching code in the rerouting driver, and inserting a template jump in the network driver interface to a template in the rerouting driver.

10. The machine-readable medium of claim 9 wherein the template jumps are inserted in the network driver interface so that a CALL instruction to the protocol driver is replaced with a JUMP to the template in the rerouting driver, the template containing the CALL instruction.

11. A computer implemented method comprising:

transmitting from a remote host to a first target computer on a network an installation application and a rerouting driver;

transmitting from the remote host to the first target computer a command to cause the first target computer to execute the installation application;

the first target computer, responsive to receipt of the command, executing the installation application, wherein the first target computer includes a network driver interface that provides for communication between one or more media access control units and one or more protocol drivers according to a set of bindings; and

the first target computer, responsive to executing the installation application, causing the modification of the network driver interface to insert the rerouting driver into

the one or more communication paths provided by the set of bindings without restarting the first target computer.

12. The computer implemented method of claim 11 wherein the modification of the network driver interface is by static patching.

13. The computer implemented method of claim 12 wherein the static patching further comprises inserting template jumps from the network driver interface to templates in the rerouting driver.

14. The computer implemented method of claim 13 wherein the template jumps are inserted in the network driver interface so that a CALL instruction to the protocol driver is replaced with a JUMP to the template in the rerouting driver, the template containing the CALL instruction.

15. The computer implemented method of claim 11 wherein the modification of the network driver interface is by dynamic patching.

16. The computer implemented method of claim 15 wherein the dynamic patching further comprises establishing a new binding between at least one media access control unit and dynamic patching code in the rerouting driver, and inserting a template jump in the network driver interface to a template in the rerouting driver.

17. The computer implemented method of claim 16 wherein the template jumps are inserted in the network driver interface so that a CALL instruction to the protocol driver is replaced with a JUMP to the template in the rerouting driver, the template containing the CALL instruction.

18. A computer system comprising:

a protocol driver;

a media access control unit;

a network driver interface to store a first binding defining a communication path between the protocol driver and the media access control unit, the network driver interface coupled to communicate packets with the media access control unit, the network driver interface patched to communicate the packets with a rerouting driver; and

the rerouting driver being coupled to communicate the packets with the protocol driver.

19. The computer system of claim 18, the rerouting driver further comprising static patching code.

20. The computer system of claim 18, the rerouting driver further comprising dynamic patching code.

21. The computer system of claim 18, the rerouting driver further comprising a capture unit to store in a buffer one or more of the packets for evaluation.

22. The computer system of claim 21, the network interface to also store a second binding defining a communication path between the rerouting driver and the media access control unit; and, the capture unit to store in the buffer the packets destined for the rerouting driver.

23. A rerouting driver for remotely installing network drivers and software in a computer system without restarting the computer system following installation, the computer system having an operating system in which a network driver interface provides communication of information between at least one media access control unit and at least one protocol driver on the computer system, the rerouting driver comprising:

control code, for controlling the rerouting driver;

binding code, for establishing at least one binding at the network driver interface so that the rerouting driver is bound to at least one media access control unit;

patching code, for inserting template jumps into at least a first section of code in the network driver interface, the template jumps providing jumps to templates in the rerouting driver so that information from at least one media access control unit destined for at least one protocol driver is rerouted to the rerouting driver;

at least one template, for receiving information from at least one template jump in the network driver interface;

inserted code, for evaluating rerouted information received by the template jumps.

24. The rerouting driver of claim 23 wherein the control code identifies a starting memory address of the network driver interface instruction code and disables access to the first section of code, and further wherein the patching code, following the disabling of access, operates to overwrite the first section of code and additional pre-determined memory addresses so that all the pre-determined memory addresses are patched.

25. The rerouting driver of claim 23 wherein the patching code responsive to receipt of information being sent from the network driver interface, determines the instruction code address that sent the information and overwrites the first section of code at that address so that memory addresses are incrementally patched as information is received from the network driver interface.

26. A method for disabling and re-enabling access to code in a multiprocessor system having a shared memory and a network driver interface comprising:

selecting a first section of code in a first central processing unit that is to be modified;

writing the first section of code into the cache memory of the first central processing unit;

overwriting a portion of the first section of code in cache memory with blocking code to create a first version of code;

writing the first version of code into shared memory;

modifying the first version of code in the cache memory to create a second version of code, wherein a portion of the code following the blocking code is overwritten with template jumps to effect a static patch of the network driver interface;

writing the second version of code into shared memory;

modifying the second version of code in the cache memory with code to create a third version of code, wherein the blocking code is overwritten to remove the blocking code; and

writing the third version of code into shared memory.

27. The method of claim 26 wherein the first section of code is located in the network driver interface.

28. A machine-readable medium having stored therein instructions, which when executed, cause a set of one or more processors to perform the following:

disabling access to a first section of code, the first section of code to be executed when to provide a communication path between a media access control unit and an application, the first section of code including a generic call; and

overwriting the first section of code with a second section of code whose

execution causes execution flow to be rerouted to a third section of code in a rerouting driver, the second section of code being no larger than the first section of code, the third section of code, when executed, completing the communication path and returning execution flow, the third section of code including additional code not present in the first section of code that is now inserted into the communication path.

29. The machine-readable medium of claim 28 wherein the second section of code contains a template jump to a template in the third section of code.

ADD A1